UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

LEVEL II

12

AD A103638

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER 16352.9-M	2. GOVT ACCESSION NO. AD-A103638	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Procedures for Heuristic Scheduling Under Limited Resources in Activity Networks		5. TYPE OF REPORT & PERIOD COVERED Technical rept.
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) S. E. Elmaghraby, Z. M. Naman		8. CONTRACT OR GRANT NUMBER(s) DAAG29-79-C-0211, NSF ENG 78-1...
9. PERFORMING ORGANIZATION NAME AND ADDRESS North Carolina State University, Raleigh, NC 27650		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS U. S. Army Research Office, Post Office Box 12211, Research Triangle Park, NC 27709		12. REPORT DATE Jun 81
		13. NUMBER OF PAGES 25
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) ARO 12 21		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

NCSU-OR-178

DTIC ELECTE SEP 2 1981 D

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

NA

18. SUPPLEMENTARY NOTES

The view, opinions, and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy, or decision, unless so designated by other documentation.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

computer programs
scheduling
operations research
resource allocation
heuristic scheduling

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This is the documentation of a program that permits the analyst to choose any one of three priority rules to schedule activities under conditions of limited resources. The program is written in FORTRAN. We give the logic of the programs, a flowchart representation, and a printout of the instructions (upon request). Also, a solved sample problem is included. This program is available at a nominal fee from the Graduate Program in Operations Research, NC State University, PO Box 5511, Raleigh, NC 27650.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED 408802

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

DTIC FILE COPY

OPERATIONS RESEARCH

NORTH CAROLINA STATE UNIVERSITY AT RALEIGH

PROCEDURES FOR HEURISTIC SCHEDULING UNDER LIMITED RESOURCES IN ACTIVITY NETWORKS

S.E. Elmaghraby and Z.M. Naman

OR REPORT NO. 178 JUNE 1981

Classification Code: Activity Networks
Scheduling

This research was partially supported by NSF Grant No. ENG-7817936 and ARO Contract No. DAAG29-79-C-0211.

PROCEDURES FOR HEURISTIC SCHEDULING UNDER LIMITED RESOURCES IN ACTIVITY NETWORKS

Abstract

This is the documentation of a program that permits the analyst to choose any one of three priority rules to schedule activities under conditions of limited resources. The program is written in FORTRAN. We give the logic of the programs, a flowchart representation, and a printout of the instructions (upon request). Also, a solved sample problem is included.

This program is available at a nominal fee from the Graduate Program in Operations Research, N. C. State University, P. O. Box 5511, Raleigh, NC 27650.

PROCEDURES FOR HEURISTIC SCHEDULING UNDER LIMITED RESOURCES IN ACTIVITY NETWORKS

S. E. Elmaghraby and Z. M. Naman

This package consists of two programs:

1. SETUP: To setup the variables in a form readable by HSP (heuristic scheculing procedure)
2. HSP: To schedule the activities prepared by SETUP.

I. SETUP

This program is to set up the variables needed for the scheduling program HSP, namely the network, activities' duration, resource consumption and resources availabilities. The following are its options:

GN:	GN=1	The network is randomly generated
	GN=2	The network is read in
GRAND:	GRAND=1	The network parameters are randomly generated.
	GRAND=2	The network parameters are read in
CONV:	CONV=1	Convert the network to A-on-N.
	CONV=2	The network is A-on-N.

Any combination of the above options could be used. For example,

GN=1 , GRAND=2 , CONV=1

means: To generate the network randomly;

The network parameters are read in; and

The network is A-on-A, so convert it to A-on-N.

<u>Input Sequence</u>

```
1.  Options:  GN, GRAND, CONV

2.  Number of nodes, number of arcs

3.  IF the network is read THEN
        IF  it is A-on-A THEN give a list of pairs (Node i,
            Node j) to represent arcs in the network
      ELSE  [The network is A-on-N] give the upper triangular
            matrix without diagonal (the adjacency matrix)
            which is a (0,1) matrix given row by row.
      ELSE  Generate the network randomly;

4.  Read the number of resources

5.  IF  the network parameters are randomly generated THEN give
        the following information:
        1.  L3:     Maximum desired activity duration
        2.  N3:     Maximum desired resource requirement
        3.  F > 1:  Resource allocation factor
        [To get the resource allocation we find the average
         resource used, then we multiply it by F.
        Average resource = Total resource/critical path.]
  ELSE  [The parameters are read in] give the following
        information:
        1.  Duration of each activity
        2.  Resource requirements to be given as FOR each
            activity
```

```
    FOR each resource
        R1:  Resource required
        D1:  Duration of this request
    END
END;
3.  Resource availability for each resource
    END
    Example:   1,  1,  1   [option card] [generate the
                           network, generate the parame-
                           ters, convert to A-on-N]
               20, 60      [network specification card]
                           [20-nodes and 60-arcs]
               3           [number of resources card]
                           [3-resources]
               15, 10, 1.5 [parameters generated informa-
                           tion] [L3=15, N3=10, F=1.5]
```

Intermediate Output:

1. Provided or generated information
2. Network as A-on-N (arc list)
3. Number of activities
4. Upper bound on project duration
5. Lower bound on project duration
6. Activity data
7. Resource availability levels.

Program Limitations:

1. Maximum number of resources = 4
2. Maximum number of activities = 99
3. Maximum sum of durations = 350

To increase the above variables, the program DIMENSION must be changed.

4. To add the network generator, do the following:
   a. Replace the comment card which indicates the position of the network generator by calling card.
   b. Take out all the "c" at Column 1 in the next Do-loop.
   c. Add the arrays NE and NS in the DIMENSION statement.
5. For the random number generator, please look at the note given in "NETWORK GENERATION" program description, OR Report No. 179, June 1981, p. 4.

II. HSP

The project scheduling problem is to assign start times to all the activities in A. Each of these start times must respect the precedence and resource constraints described above. Another assumption made in this study is that of non-preemptive scheduling: Once started, an activity must continue without interruption for its duration. As well as respecting all constraints, schedulers attempt to assign start times so that some objective is achieved.

The objective of the scheduling algorithm presented below is to minimize the project duration. Define the set, A(i), to be the set of activities immediately succeeding i. By L we denote the set of activities having no successors. Activity i is in L if A(i) is the empty set. Project duration is then obtained as the $\max_{i \in L} (t(i) + d(i) - 1)$. Necessarily, the maximization of total resource utilization is accomplished by minimizing project duration. Moreover, minimizing project duration also assures the minimization of delay penalties incurred per period beyond the (resource free) critical path length. Finally, the chosen objective is applicable to a variety of situations. For example, a capital intensive project, such as a large construction project, might require substantial financing to secure the necessary resources. Interest payments on the borrowed capital cut into the constructor's profit until the project is completed and the resources are no longer needed. Minimizing project duration thus minimizes such interest costs.

Heuristics Employed

In the heuristic scheduling procedure, an adapted _binpacking rule_ is used to assign the activities to the resource units. The _first fit decreasing_ (FFD) binpacking rule is one of the most popular scheduling rules. It has also been applied with some success to the problem of memory storage allocation by some computer scientists. The FFD rule specifies the packing of different sized objects into containers or bins. The objects are first ordered by decreasing size. The largest object is then assigned to the first container in which it will fit. The second largest object is then assigned to the first container in which it will fit, etc. In the project setting, the objects are the jobs and the containers are the resources. There are as many "containers" as there are different resources, with each job "occupying" all containers simultaneously. This is a multi-dimensional binpacking problem which we wish to avoid by adopting a simple rule for "sizing" the jobs. We address this "sizing" problem next.

For project scheduling, it is clear that the limited availability of the various resources may cause an activity to be delayed. Such delays can result in project completion delay. A measure of the delay in project completion caused by delaying an activity should serve well as an activity's "size."

The original idea for measuring the project delay caused by delaying activity i was to sum the durations of all activities succeeding activity i in the precedence ordering plus the duration of activity i itself. The larger the priority number ("size") of activity i, p(i), the longer the expected project delay should be, if activity i is delayed. To reduce the computational burden, we adopted a slightly different priority number:

$$p(i) = \begin{cases} d(i) & \text{if } A(i) = \phi, \\ d(i) + \sum\limits_{j \varepsilon A(i)} p(j), & \text{if } A(i) \neq \phi. \end{cases}$$

These numbers can be assigned in one pass through the precedence ordering. However, the recursion allows the counting of some durations more than once in determining the priority number for certain activities.

The two other priority rankings available in this package are: the <u>longest remaining path</u> (LRP) heuristic and the <u>shortest imminent activity</u> (SIA) heuristic. LRP gives a higher priority to an activity with a longer critical subpath (to project completion) than a competing activity. The SIA heuristic mimics the shortest processing time rule from job shop scheduling. Characteristic of both schemes, and a number of other heuristics, is the neglect of the total impact on project completion that delaying an activity can produce. The impact of delaying an activity can affect all its successors. The proposed priority ranking is an attempt to reflect the full impact of delaying an activity on the project duration by adopting a simple surrogate measure.

To sum up, the three heuristic "sizing" priority rules are:

CR1: The sum of durations of all succeeding activities

CR2: The longest path to project termination (LRP)

CR3: The shortest imminent (available) activity (SIA)

The scheduling process begins by forming the set, CS, of all activities where predecessors have been scheduled prior to the current scheduling period. On the first iteration, CS will consist of all initial activities (i such that $B(i) = \phi$). In general CS will contain activities that are unordered with respect to precedence. The adopted FFD rule assigns activities of CS to available resource units. The activities are the items, ranked by decreasing "size" (priority number). The "size" of each bin in a particular scheduling period will be the number of available units of that resource type. The activities are checked for fit starting with that activity whose priority number is the largest in CS. If activity i "fits", then activity i is scheduled to begin on period T1. After updating resource availabilities, the next activity is checked for fit, etc., until all activities in CS have been considered for scheduling at T1, where T1 is the current scheduling period.

The algorithm is iterative in nature. CS and T1 are updated after each iteration. The process repeats until all activities are assigned. The updated scheduling period is determined by the completion times of the activities scheduled during the previous iteration. It is the earliest completion time of any of the activities scheduled in the previous iteration.

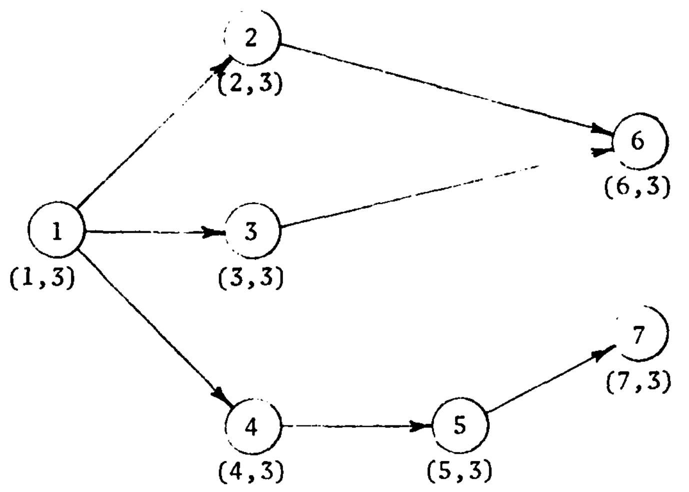
$$T1_{new} = T1_{old} + \min_{i \in CS_{old}} (M(i, T1_{old}) \cdot d(i)),$$

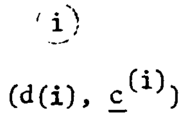
where d(i) is the duration of activity i, and

$$M(i,T1_{old}) = \begin{cases} 1 & \text{if activity } i \text{ started on } T1_{old}, \\ \infty & \text{if } i \text{ not started on } T1_{old}. \end{cases}$$

The intention in skipping over the intervening periods between $T1_{old}$ and $T1_{new}$ is to avoid wasting time attempting to schedule activities when few resources are available in the scheduling period. Upon completion of an activity, resource units are "released" and other activities can perhaps be undertaken. A flowchart of HSP logic is shown in Fig. 1.

An Example





The figure above is a graphical representation of the precedence relationship among the seven activities of an example project. An arc, (i,j) implies that activity j cannot be started until activity i has been completed. For example, activities two, three, and four cannot be started until activity one has been completed. Assume that activity durations, d(i) = i, for i = 1,2, ..., 7. Also, one resource type is required in quantities of three units per day by each activity ($\underline{c}^{(i)}(j) = 3$, for j = 1,2, ..., d(i)). Six units of the resource are available for each of the scheduling periods ($\underline{r}(t) = 6$, for t = 1,2, ..., $\overline{T}$ where $\overline{T}$ is an upper bound on project duration).

First, the priority numbers are assigned in one backward pass through the precedence network. Suppose we adopt priority rule CR1. Activities six and seven are the terminal activities of this project (i.e., L = {6,7}). Thus their priority numbers are just their durations (p(6) = d(6) = 6, p(7) = d(7) = 7). Priorities for the immediate predecessors of six and seven, activities two, three, and five, can now be computed. Their priorities are the sum of activity duration and the priority numbers of the successors: p(2) = d(2) + p(6) = 2 + 6 = 8, p(3) = d(3) + p(6) = 3 + 6 = 9, p(5) = d(5) + p(7) = 5 + 7 = 12. Activity four has priority p(4), given by the sum of its duration and the priority number of its successor, activity five. Thus p(4) = d(4) + p(5) = 4 + 12 = 16. Finally, activity one has as successors activities two, three, and four. Thus p(1) = d(1) + p(2) + p(3) + p(4) = 1 + 8 + 9 + 16 = 34. Now that activity priorities have been computed, the scheduling process can begin.

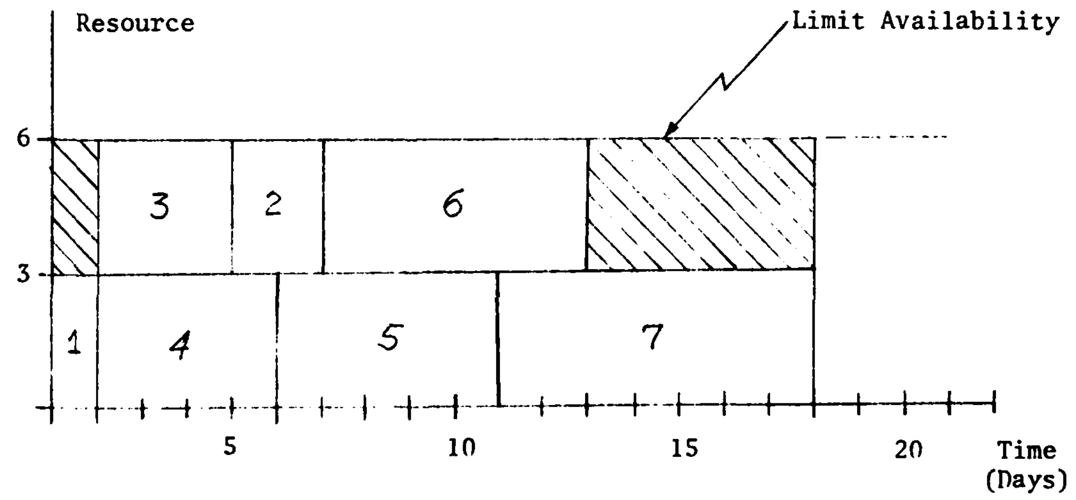
On the first scheduling iteration, CS will consist of activity one since it is the lone initial activity in the project. Three units of the resource are required by activity one, and six units are available on day one. Thus activity one is scheduled for day one.

Day two is the updated scheduling period on the second iteration since activity one is completed on day one. CS will consist of activities two, three, and four, the immediate successors of activity one. The scheduling priority is four (p(4) = 16), three (p(3) = 9), and two (p(2) = 8). Activity four is first checked for fit. Three units of the resource are required on days two through five, and six units are available. Since activity one is completed by day two, activity four is scheduled to start on day two. Now only three resource units remain available on days two through five. Next, activity three is checked for fit. Three resource units are required by activity three on days two through four, and three units are available. Activity one is completed by day two, so activity three is scheduled to start on day two. Thus no units of the resource are available on days two, three, and four. Activity two is checked for fit next. But, with no resources available and three units required, activity two cannot be scheduled on day two. So the set CS and scheduling period are updated and we proceed to the third iteration.

The new set of activities to be scanned will include activity two that was not scheduled on the previous iteration and activity five since its predecessor has now been scheduled. The new scheduling period is day five since, of the two activities scheduled on the previous iteration, activity three finishes sooner (on day four) releasing some resources. Activity five has a larger priority number than activity two. Hence activity five is considered for scheduling before activity two. However, activity four is still in progress on day five so activity five is ineligible for scheduling on day five. Next, activity two is considered for scheduling on day five. Three units of the resource are required by activity two on days five and six. Three units are available on day five, and six units are available on day six. No precedence constraint is violated, so activity two is scheduled to begin on day five. With that assignment, no resource units remain available on day five and three units remain available on day six. Both activities in CS, at this iteration, have been checked. We proceed to update CS and the scheduling period for the next iteration. The updated CS consists of activities five and six since the scheduling of activity two releases activity six for consideration. The new scheduling period will be day six since activity four is completed on day five.

On the fourth iteration, activity five has priority and is scheduled on day six. Available resource units are updated and activity six is considered. Sufficient resources are available, and no precedence constraint is violated in starting activity six on day seven. Hence, that assignment is made.

Finally, activity seven is considered on the fifth iteration. The updated scheduling period is day eleven (given by the completion of activity five). No resource or precedence constraints are violated by starting activity seven on day eleven so that assignment is made. The resource "skyline" or profile chart below illustrates the example project's schedule. The project is completed on day seventeen. This duration coincides favorably with the project's critical path length in the absence of resource constraints (seventeen days), and is therefore optimal.



(Shaded areas represent idle resource-time)

The project scheduling algorithm described above is an analog of the heuristic algorithm for the ALB problem called the "Ranked

Positional Weight" heuristic, which was proposed by Hegelson and Birnie in 1961. An extension of this heuristic was suggested by Mansoor in 1964.

Advantages of Heuristic Scheduling

A significant contribution of heuristic scheduling might be that of increasing the computational efficiency of optimal project scheduling procedures. One attempt might be to link the proposed scheduling algorithm (or any other good heuristic algorithm) with a branch and bound algorithm. For such a tandem algorithm, the size of the space to be searched will be smaller than the search space of a straightforward branch and bound procedure.

We have noted that a lower bound on project duration is the critical path length. Similarly, we obtain a lower bound on the start time of an activity by computing the earliest start schedule (ESS). The ESS neglects any resource constraints and starts an activity as soon as all precursory activities are completed:

$$ESS(i) = \begin{cases} \max\limits_{j \in B(i)} (ESS(j) + d(j)), & \text{if } B(i) \neq \phi , \\ 1 \quad \text{if } B(i) = \phi . \end{cases}$$

If a tight upper bound on optimal start time is available for all activities, then the chance of finding an optimal schedule by a branch and bound scheme is greatly improved. A good heuristic algorithm will provide such a bound at a small price.

The heuristic's schedule yields a feasible project duration, T. The late start schedule, LSS_T, analogous to the early start schedule described above, is computed:

$$LSS_T(i) = \begin{cases} \min\limits_{j \in A(i)} (LSS_T(j) - d(i) + 1), & \text{if } A(i) \neq \phi, \\ T - d(i) + 1, & \text{if } A(i) = \phi. \end{cases}$$

The optimal start time(s) of activity i is(are) now bounded: $ESS(i) \leq T_{opt}(i) \leq LSS_T(i)$. Activity i cannot be started before its predecessors are completed (ESS(i)) and an optimal schedule cannot start activity i after its late start time with project duration T since $T_{opt} \leq T$.

Thus the search tree of the branch and bound procedure is pruned. The tightness of these bounds and hence the reduction of the search space is increased as the heuristic schedule is improved. Therefore, an efficient, dependable heuristic scheduling **algorithm** might play a crucial role in increasing the computational reliability of an analytical procedure for the project scheduling problem.

The following information is needed to be passed from SETUP:

1. Number of resources
2. Number of activities
3. Upper bound of planning horizon
4. Network in A-on-A
5. Availability of each resource
6. Requirement of each resource/activity/time period

7. Duration of each activity
8. Index vector K
9. Early start schedule

Options

P7:	P7=1	Do not print iteration information.
	P7=2	Print iteration information.
SCH:	SCH=1	Scheduling using CR1
	SCH=2	Scheduling using CR1 and CR2
	SCH=3	Scheduling using CR1 and CR3
	SCH=4	Scheduling using CR1, CR2, and CR3

For example, P7=2

SCH=3

imply: Print iteration information.

Use CR1 and CR3 as priority criteria.

Input Sequence

P7, SCH [on the same card]

Output

1. For each scheduling iteration (optional)
   a. Current scanned subset of activities and their priority numbers
   b. Activities delayed because of resource limited availability
   c. Activities delayed because of precedence
   d. Activities scheduled
2. Project table

   For each activity Print:

   Duration, Priority number, HSP-start, Early-start

3. Resource utilization

   For each resource Print:

   a. Time, availability, utilization, leftover

   b. Draw the skyline for the resource utilization *vs*. time.

4. Adjusted percent resource utilization, for all resources

5. Lower bound on duration

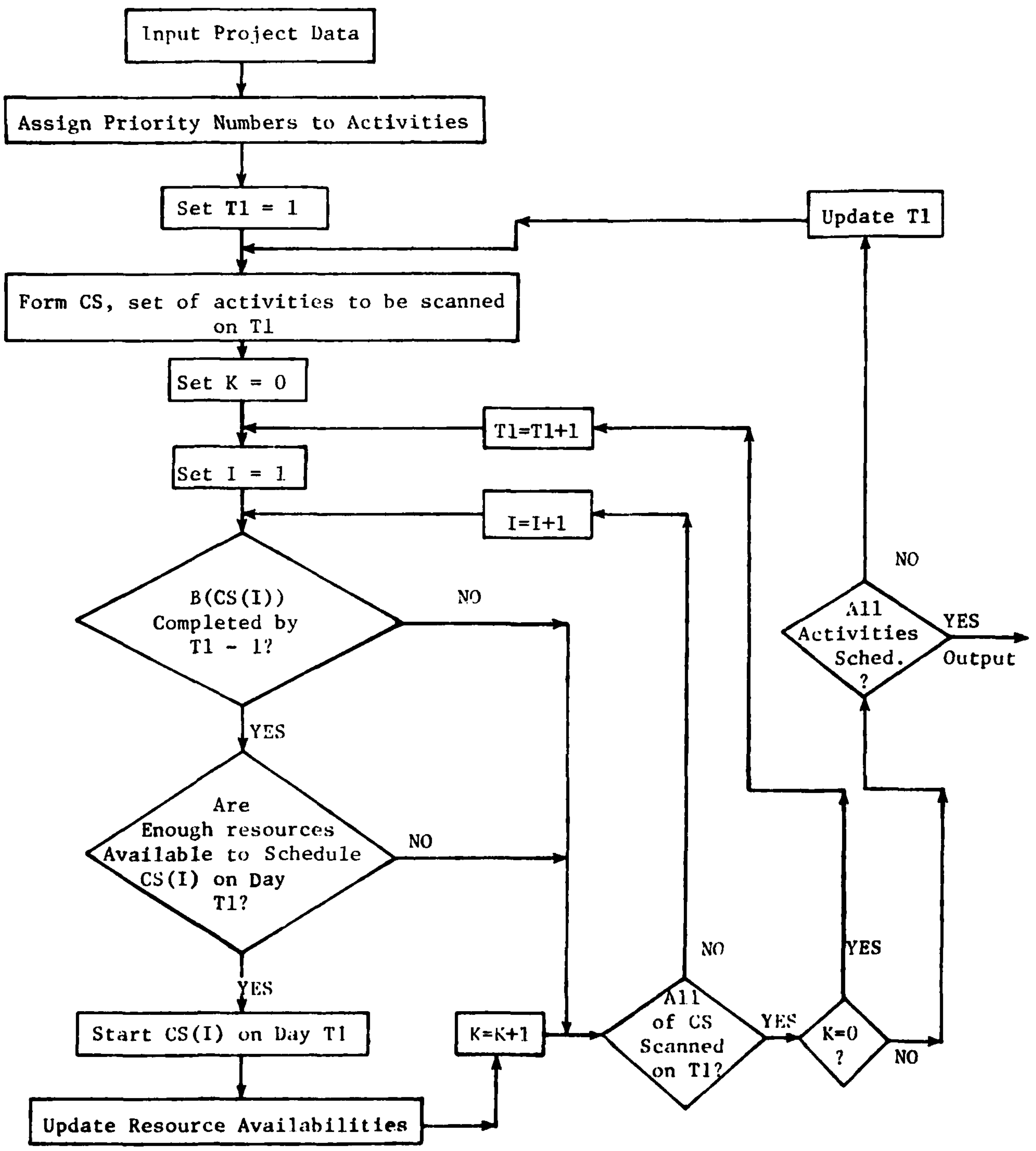
6. Completion time from HSP

7. Upper bound of duration

Limitation

Please see the limitation given in SETUP.

Figure 1. Heuristic Project Scheduling Flowchart


Input Project Data
Assign Priority Numbers to Activities
Set T1 = 1
Update T1
Form CS, set of activities to be scanned on T1
Set K = 0
T1=T1+1
Set I = 1
I=I+1
NO
B(CS(I)) Completed by T1 - 1?
NO
All Activities Sched. ?
YES
Output
YES
Are Enough resources Available to Schedule CS(I) on Day T1?
NO
NO
YES
YES
Start CS(I) on Day T1
K=K+1
All of CS Scanned on T1?
YES
K=0 ?
NO
Update Resource Availabilities

1. (OPTION) To read the options for this RUN. Read all the necessary information depending on the option that is given.
   P = Number of nodes if A-on-N
     = Number of arcs if A-on-A
   K = An array to tell where each resource requirement is started for each activity.

```
SETUP
OPTION
READ, GN, GRAND, CONV
Print the option used in this run
READ, NARC, NODE
GN = 1
  Y:
    NETGEN
    K = 1
    FOR I = 1 to NARC
        V(K) = NE(I)
        V(K+1) = NS(I)
        K = K + 2
  N:
    CONV=1
      Y:
        NUP = 2*NARC
        READ, V(J)
        J = 1, NUP
      N:
        NUP = NODE*
        (NODE-1)/2
        J = 1, NUP
READ, RI
GRAND = 2
  Y:
    READ, L3, N3, F
    → CONVERT
  N:
READ, D(I), I = 1, P
FOR I = 1 to P
    FOR RO = 1, RI
        READ, R1, D1
        FOR DO = 1, D1
            R(DO+K(I),RO) = R1
READ, A(1,RO), RO = 1, RI
FOR I = 1, P
    B2 = B2 + D(I)
FOR RO = 1, RI
    FOR I = 2, B2
        A(I,RO) = A(1,RO)
```

2. (CONVERT) To convert A-on-A to A-on-N and create an array X which is the one dimensional representation of the upper triangle w/o diagonal matrix.

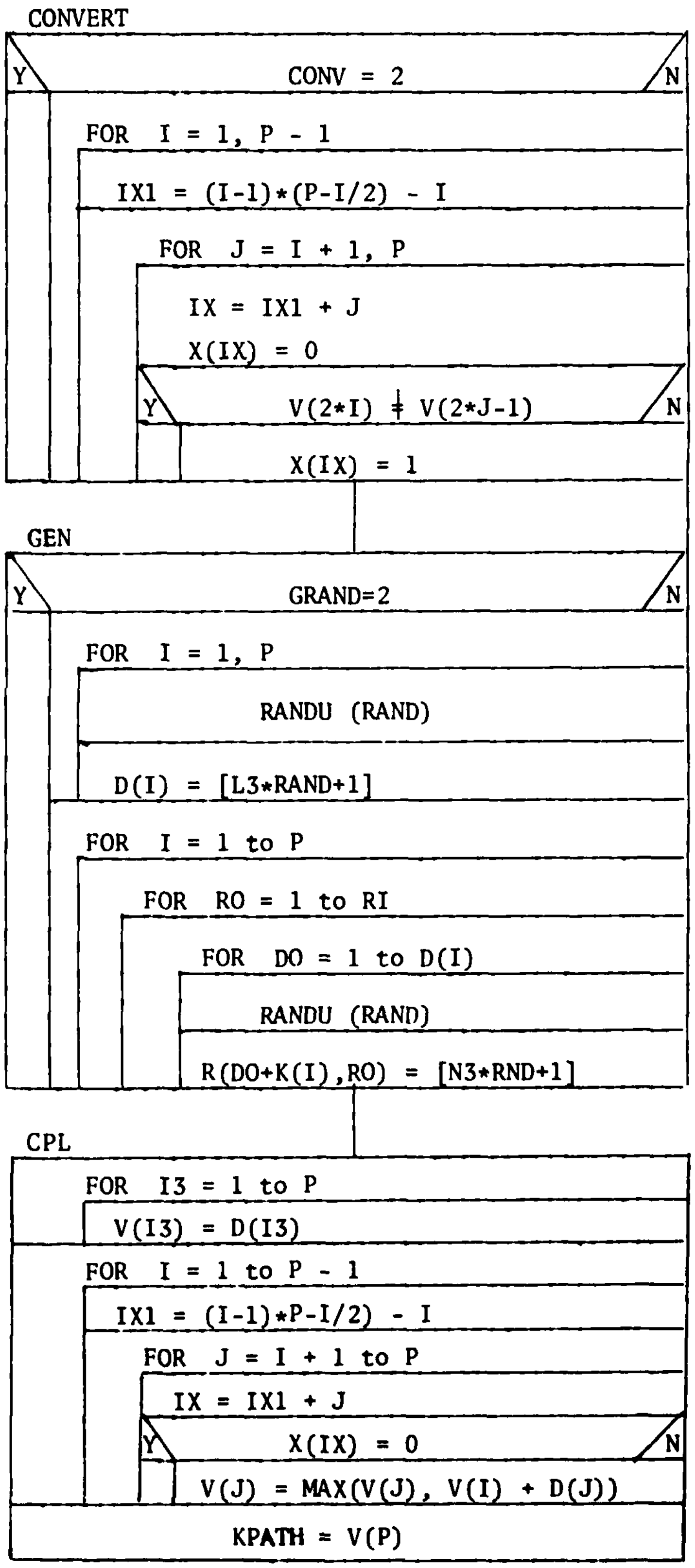
   X = 1 if activity I precedes J
     = 0 otherwise

3. (GEN) To generate randomly the network parameters.

   D(I) - duration of each activity

   R - resource requirement for each activity at each time period for each resource.

4. (CPL) To calculate the critical path of the network. The early start schedule could be produced by E(J) = V(J) - D(J) + 1; $1 \le J \le P$ where V is the vector generated by CPL.



```
CONVERT
IF CONV = 2 (Y / N)
  Y:
    FOR I = 1, P - 1
        IX1 = (I-1)*(P-I/2) - I
        FOR J = I + 1, P
            IX = IX1 + J
            X(IX) = 0
            IF V(2*I) ≠ V(2*J-1) (Y / N)
              Y:
                X(IX) = 1

GEN
IF GRAND=2 (Y / N)
  Y:
    FOR I = 1, P
        RANDU (RAND)
        D(I) = [L3*RAND+1]
    FOR I = 1 to P
        FOR RO = 1 to RI
            FOR DO = 1 to D(I)
                RANDU (RAND)
                R(DO+K(I),RO) = [N3*RND+1]

CPL
FOR I3 = 1 to P
    V(I3) = D(I3)
FOR I = 1 to P - 1
    IX1 = (I-1)*P-I/2) - I
    FOR J = I + 1 to P
        IX = IX1 + J
        IF X(IX) = 0 (Y / N)
          Y:
            V(J) = MAX(V(J), V(I) + D(J))
KPATH = V(P)
```

5. (AVL) To generate the resource availability by getting the average requirement and multiplying it by the factor F. Check if there is any request higher than this suggested value.
6. (PRINT) Print and tabulate the data that will be passed to HSP.

```
AVL
IF GRAND=2
  Y:
  N:
    FOR RO = 1 to RI
        A1 = 0
        FOR PO = 1 to P
            FOR DO = 1, D(PO)
                A1 = A1 + R(K(PO) + DO, PO)
        A3 = [A1*F/KPATH]
        FOR PO = 1 to P
            FOR DO = 1 to D(PO)
                A3 = MAX(A3, R(DO+K(PO), RO))
        FOR TO = 1 to B2
            A(TO,RO) = A3

PRINT
    Print the data that will be passed to the
    scheduling program
RETURN
```

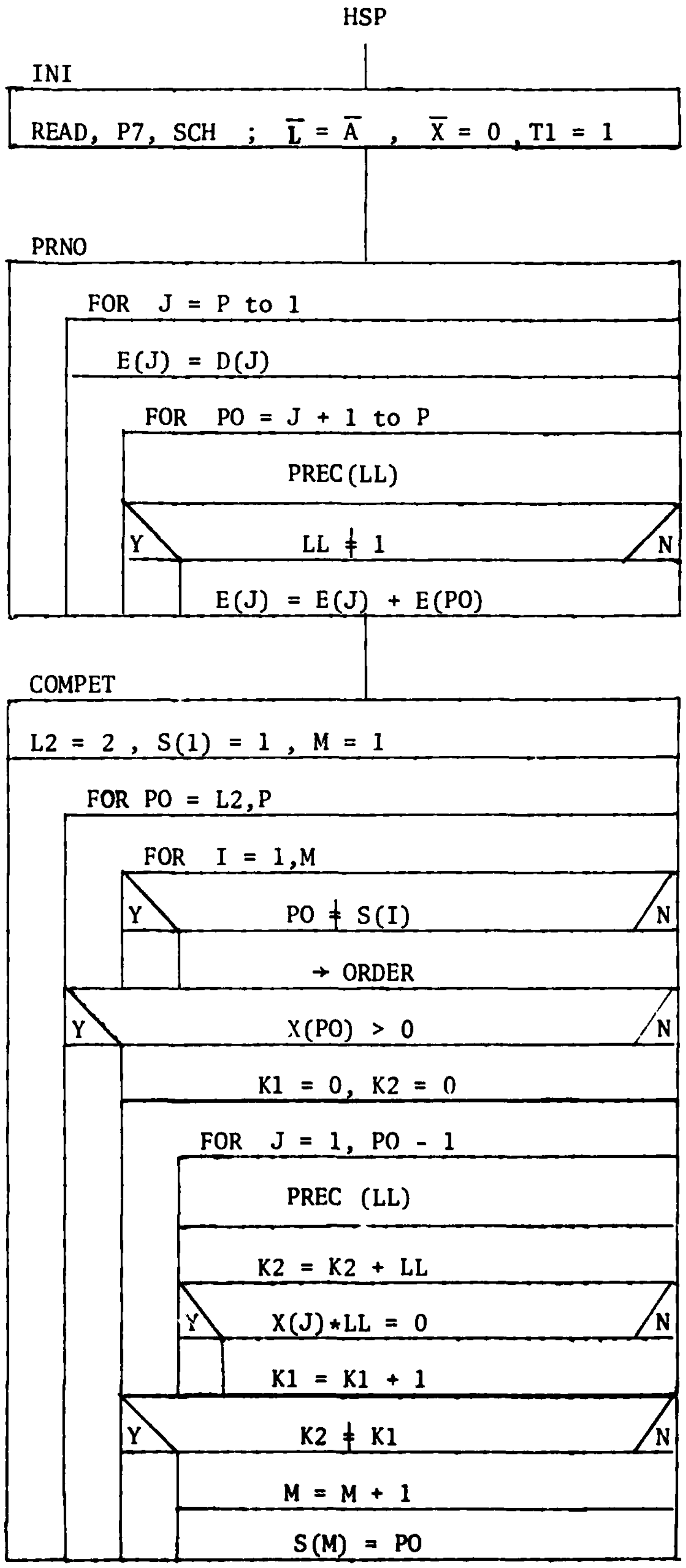
1. (INI) Read option variables P7,SCH. Set the matrix L = matrix A. Set vector X to zero.
2. (PRNO) Compute the priority number by CR1. Store it in vector E.
3. (COMPET) Get a set of simultaneous activities. Save the candidates in vector S.

   X = 0 if the activity has not been scheduled

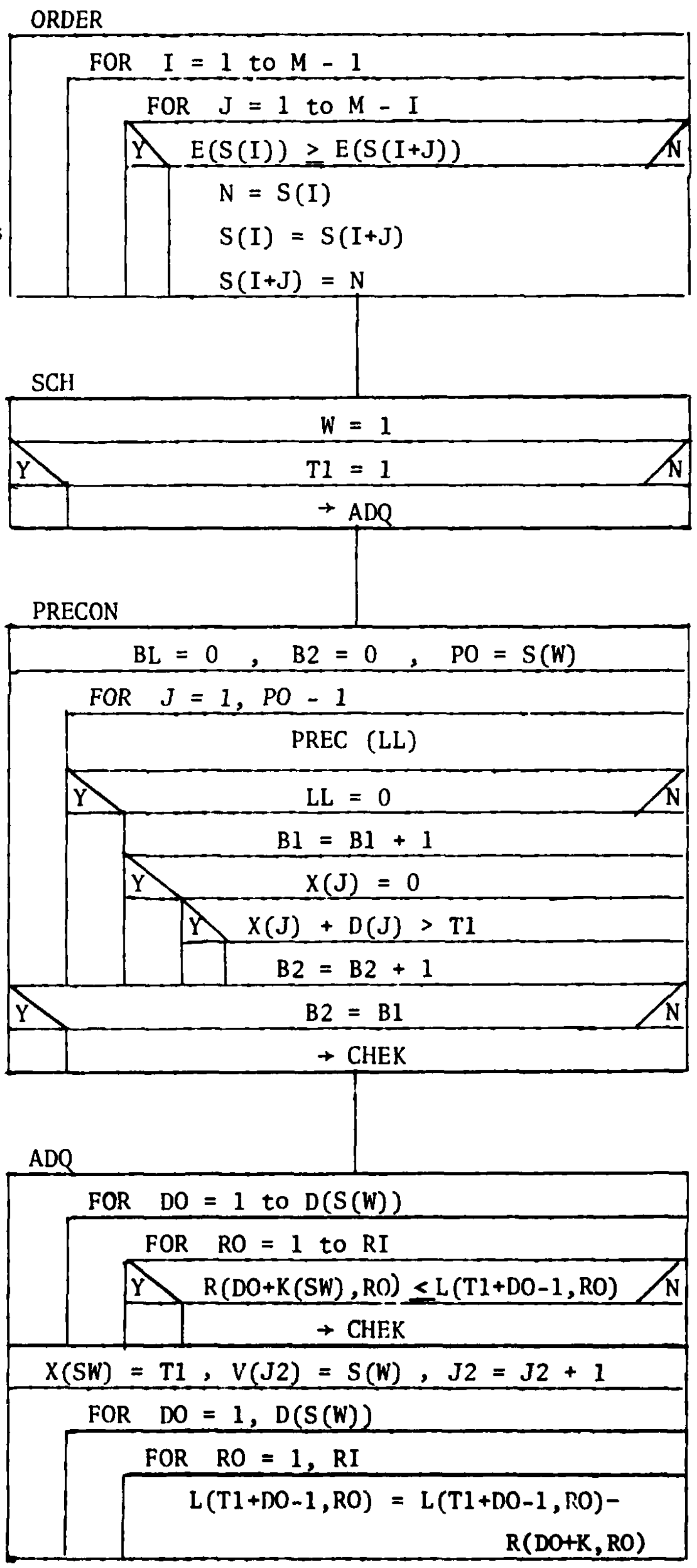
   X = T1 the time the activity is scheduled

   LL = 0 No event between activities PO and J

   LL = 1 Otherwise



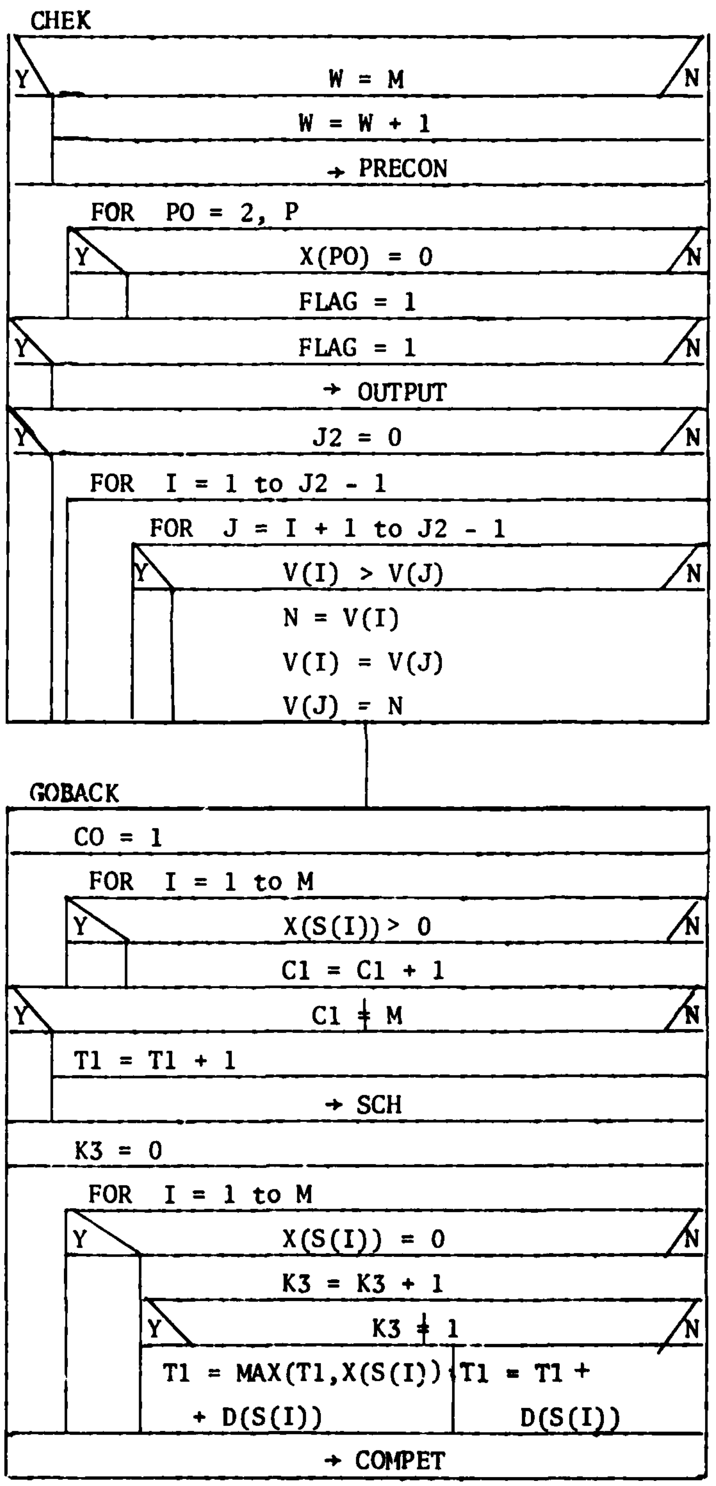
4. (ORDER) Sort the set S according to their priority number.
5. (SCH) The beginning of scheduling the set S.W- will keep track of those activities that are scheduled or cannot be scheduled because of precedence constraints. T1 is the time counter.
6. (PRECON) To check if there is any precedence constraints. <u>IF</u> there is none <u>THEN</u> check for resource constraints. <u>ELSE</u> go to CHEK.
7. (ADQ) To check if the resources requested by activity S(W) is available. <u>IF</u> yes <u>THEN</u> schedule that activity, and update the resource available. <u>ELSE</u> go to CHEK.



```
ORDER
FOR I = 1 to M - 1
    FOR J = 1 to M - I
        Y  E(S(I)) ≥ E(S(I+J))  N
            N = S(I)
            S(I) = S(I+J)
            S(I+J) = N

SCH
W = 1
Y  T1 = 1  N
    → ADQ

PRECON
BL = 0 , B2 = 0 , PO = S(W)
FOR J = 1, PO - 1
    PREC (LL)
    Y  LL = 0  N
        B1 = B1 + 1
        Y  X(J) = 0
            Y  X(J) + D(J) > T1
                B2 = B2 + 1
Y  B2 = B1  N
    → CHEK

ADQ
FOR DO = 1 to D(S(W))
    FOR RO = 1 to RI
        Y  R(DO+K(SW),RO) ≤ L(T1+DO-1,RO)  N
            → CHEK
X(SW) = T1 , V(J2) = S(W) , J2 = J2 + 1
FOR DO = 1, D(S(W))
    FOR RO = 1, RI
        L(T1+DO-1,RO) = L(T1+DO-1,RO)-
                              R(DO+K,RO)
```

8. (CHEK) To check if all the candidate activities are checked. <u>IF</u> they are, <u>THEN</u> check if any activity has not been scheduled. <u>IF</u> there are any, <u>THEN</u> order the already scheduled activities and prepare to go to GOBACK. <u>ELSE</u> go to OUTPUT. <u>ELSE</u> go to PRECON.

9. (GOBACK) To prepare the variables for going back either to schedule whatever activities are left in S or to find a new competing set of activities. Update the time clock.



10. (OUTPUT) To print the calculated schedule.

PREC - A block used to calculate the vector image of the adjacency matrix. Then pass LL.

MAIN - The main program that calls the subroutines SETUP and HSP.

```
OUTPUT
    Print the output tables and
    draw the sky line for each resource
    RETURN
```

```
PREC
    IX1 = (J-1)*(P-J/2) - J + PO
    LL = XB(IXI)
    RETURN
```

```
MAIN
    SETUP
    HSP
    END
```

List of Works Consulted

Davis, Edward W., (1973), "Project Scheduling Under Resource Constraints - Historical Review and Categorization of Procedures," I.E. Transactions, 5(4), 297-313.

Elmaghraby, S.E., (1977), Activity Networks, Wiley, New York.

Hegelson, W.B. and Birnie, D.P., (1961), "Assembly Line Balancing Using Ranked Positional Weight Technique," J. Ind. Eng., 13(6), 394-398.

Mansoor, E.M., (1964), "Assembly Line Balancing -- An Improvement on the Ranked Positional Weight Technique," J. Ind. Eng., 15(2), 73-77.

Reingold, Nievergelt, and Deo, (1977), Combinatorical Algorithms, Theory and Practice, Prentice-Hall, Englewood Cliffs, N.J.

DATE
ILME